

A development of an accelerator board dedicated for multi-precision arithmetic operations and its application to Feynman loop integrals

S Motoki¹, H Daisaka², N Nakasato³, T Ishikawa¹, F Yuasa¹,
T Fukushige⁴, A Kawai⁴, J Makino⁵

¹ High Energy Accelerator Research Organization (KEK), 1-1, Oho, Tsukuba, Ibaraki, 305-0801, Japan

² Hitotsubashi University, 2-1, Naka, Kunitachi, Tokyo, 186-0801, Japan

³ University of Aizu, Aizu-wakamatsu, Fukushima, 965-8580, Japan

⁴ K&F Computing Research Co., 1-21-6-407, Kojimacho, Chofu, Tokyo, 182-0026, Japan

⁵ RIKEN Advanced Institute for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan

E-mail: smotoki@post.kek.jp

Abstract. Higher order corrections in perturbative quantum field theory are required for precise theoretical analysis to investigate new physics beyond the Standard Model. This indicates that we need to evaluate Feynman loop diagrams with multi-loop integrals which may require multi-precision calculation. We developed a dedicated accelerator system for multi-precision calculations (GRAPE9-MPX). We present performance results of our system for the case of Feynman two-loop box and three-loop selfenergy diagrams with multi-precision.

1. Introduction

With the discovery of Higgs particle at the CERN Large Hadron Collider, precision measurements are expected at a future International Linear Collider to explore new physics beyond the Standard Model. In tandem with experiments, an accurate theoretical prediction is required. To meet such a demand, higher order correction in perturbative quantum field theory becomes more and more important, and methods to evaluate multi-loop integrals precisely should be provided.

We have been developing DCM (Direct Computation Method) for loop integrals [1]. This is a fully numerical method of the combination of multi-dimensional integration and the extrapolation technique. It is known that the accurate numerical evaluation of the integral is a hard problem due to its divergent nature. Yuasa et al. [2] reported that the numerical evaluation is numerically unstable with double-precision operations. A solution to this difficulty is to compute the integral in multi-precision arithmetic, that is, it needs more bits for mantissa than double-precision. In addition, we have a case that needs more bits for exponent. With Double Exponential Formulas for numerical integration (DE) [3], we can estimate that 15-bit wise exponent is required to reduce a relative error smaller than a reference value, although the exponent of double-precision is only 11-bit wise.

There are several ways to accomplish multi-precision arithmetic. One way is to use the specific softwares, for examples, GMP [4], MPFR [5], ARPREC [6], MPFUN90 [6], DD/QD [6], and `quadmath` on a new version of GNU compiler. These softwares are easy to use. However, high performance can not be expected, since one multi-precision arithmetic operation is realized with, at least, more than 10 double-precision arithmetic operations in these softwares. Another way is to design a new, dedicated hardware which has circuits for calculation with precision higher than double-precision. We have been developing such a hardware based on the fundamental idea of GRAPE originally developed for gravitational N -body simulations [7]. We designed GRAPE-Multi-Precision (MP) processor which consisted of a number of processing elements (PE) and memory components with dedicated circuits for quadruple, hexuple, octuple-precision arithmetic. We implemented this processor on a structured ASIC (Application Specific Integrated Circuit) which was called GRAPE-MP [8], and on an FPGA (Field Programmable Gate Array) board with a control processor, which was called GRAPE-MPX [9]. On these systems, we measured the performance of quadruple, hexuple, octuple-precision calculation and obtained the higher performance compared to a software implementation.

In this work, we developed a new system called GRAPE9-MPX. It is similar to GRAPE-MPX but is a rather larger system. This means that larger number of PEs can be implemented in the new system, which will be suited for large scale simulations. In addition to designing a hardware, we have been developing a programming interface named Goose and LSUMP which enable us easily to use GRAPE9-MPX.

This paper is organized as follows. Next section covers a brief explanation of the design and the implementation of the new system from hardware and software viewpoints. In section 3, we briefly explain Feynman loop integrals used for the performance measurement. The results are shown in section 4. The last section is devoted to the summary and our future prospects.

2. GRAPE9-MPX system

2.1. Hardware

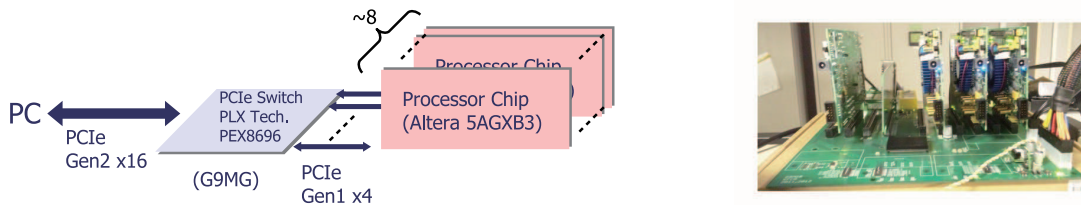


Figure 1. Schematic diagram (left) and picture (right) of GRAPE9-MPX. Note that 4 FPGA boards are included in the system.

Figure 1 shows a schematic diagram and a picture of GRAPE9-MPX system. It is an accelerator which is connected to a host computer. It consists of FPGA boards (up to 8 boards available) which are housed on a PCIe extender board (G9MG). For the FPGA board, we used Altera Arria V board from Altera Co. [10]. On this FPGA, we implemented the MP processor and its Control Processors (CP). The MP processor consists of PEs and memory components for quadruple/hexuple/octuple-precision arithmetic (named as MP4/MP6/MP8), which forms a Single-Instruction-Multiple-Data (SIMD) processor. Each PE has a floating-point multiply unit and an add unit which perform in every clock cycle. Note that for MP4, we used the numerical representation compatible with IEEE-754-2008:binary128 format. For details of the PE, CP, and the numerical representations, see Daisaka et al. [8] and Nakasato et al. [9].

One of applications suited for our system is an interaction type calculation as $f_i = \sum_{j=1}^{n_j} f(X_i, Y_j)$, where X_i is i -th element of X , Y_j is j -th element of Y , and n_j is the number of

elements of Y . The function $f(X_i, Y_j)$ describes an interaction form of X_i and Y_j . Note that data X_i is set on i -th PE, whereas data Y_j is set on all PE, then each PE calculates $f(X_i, Y_j)$ and sums up from $j = 1$ to $j = n_j$ in accordance with instructions. An example of this type calculation is gravitational interactions among particles. We should note that a multi-dimensional integration can be expressed in a similar form by *loop fusion* technique that merges multiple loops into a single loop. This indicates that we can accelerate the calculation of Feynman loop integrals effectively by using GRAPE9-MPX.

Table 1 summarizes our current implementation of MP4/MP6/MP8 processors on the FPGA board. The peak performance of MP4, for example, can be estimated as 2 (operations) \times 36 (PEs) \times 92 (MHz) = 6.6 Gflops per board. It adds up to 26.4 Gflops for the system with 4 boards.

	MP4	MP6	MP8
Number of PEs	36	20	12
Clock Speed(MHz)	92	81	70
Theoretical Peak(Gflops)	6.6	3.2	1.6

Table 1. Number of PEs, clock speed, and theoretical peak performance for MP4/MP6/MP8 implemented on the FPGA.

2.2. Programming interface

We have also developed Goose compiler which provides a programming interface for GRAPE9-MPX system. Figure 2 shows the flow of the compile process for our system, and the part of a sample program for one-loop box integral. Goose is a directive base compiler like OpenMP, in which a directive is inserted in an original code. By the directive (`#PRAGMA GOOSE PARALLEL` seen in the sample code), Goose extracts the loop next to the directive and generates an intermediate representation which is like an assembler code. Goose also generates API calls in which functions of I/O interfaces for GRAPE9-MPX are embedded. In order to convert the intermediate representation to machine instructions (kernel code) for the MP processor, Goose uses LSUMP backend [11] which is a domain specific language (DSL) compiler specially developed for the MP processor. The kernel code is read by an executable file which is generated from the API calls by C/C++ compiler. For more detail, see Nakasato [11].

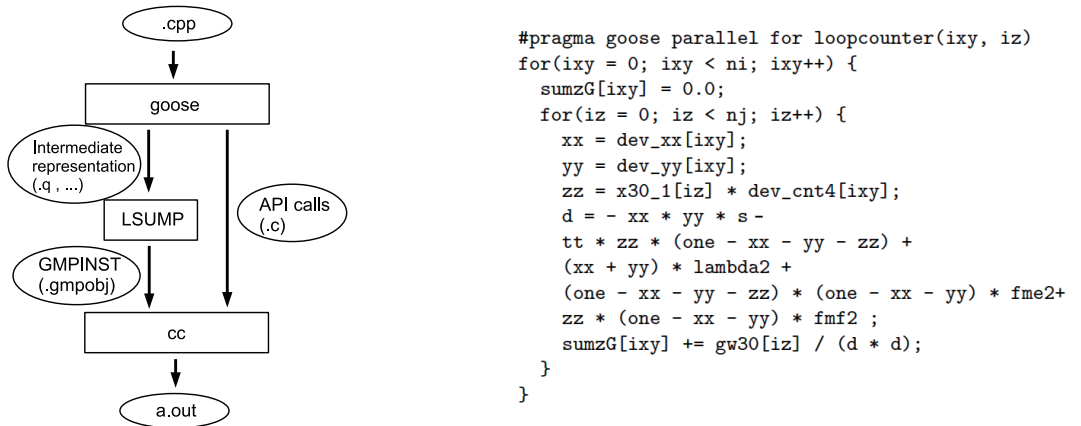


Figure 2. Flow of the compile process (left) and a sample code with the directive (right) for our system.

3. Feynman loop integrals

Figure 3 shows the two-loop crossed box diagram and the three-loop selfenergy diagram used for the performance measurement. The number of dimensions of the loop integral is six for the



Figure 3. Two-loop crossed box diagram and three-loop selfenergy diagram used for the performance measurement. In these diagrams, p_i and x_i denote a momentum of an external line and a Feynman parameter, respectively.

two-loop crossed box, and seven for the three-loop selfenergy, respectively. We consider the case where all masses of the internal lines are 1 and set the Mandelstam variables $s = t = 1$ and the external momenta $p_i^2 = 1 (1 \leq i \leq 4)$ for the two-loop crossed box diagram and $s = 1$ and $p^2 = 1$ for the three-loop selfenergy diagram, respectively. We choose these values to compare numerical results to ones in the previous studies [12, 13].

With these parameters, the integrals of the two-loop crossed box with variable transformations from $\{x_i\}$ to $\{\xi_i\}$ to change the integration domain to the unit cube from the unit simplex is expressed as

$$I_{2loop} = 2 \int_0^1 d\xi_1 d\xi_2 d\xi_3 d\xi_4 d\xi_5 d\xi_6 (\xi_1^2 \xi_1'^3 \xi_2' \xi_4' \xi_4') \frac{C}{D^3}, \quad (1)$$

where

$$\begin{aligned} C &= \xi_1' (\xi_1 + \xi_1' \xi_4 \xi_4'), \\ D &= \xi_1' ((\xi_1 + \xi_1' \xi_4 \xi_4') \\ &\quad - (\xi_1 (\xi_1 \xi_2' \xi_2' \xi_3 \xi_3' + \xi_1' \xi_4 \xi_4' (\xi_2' (\xi_3 \xi_5' \xi_6 + \xi_3' \xi_5 \xi_6') - \xi_2 \xi_5 \xi_6)) \\ &\quad + \xi_1 \xi_2 (\xi_1' \xi_4 \xi_4' (-\xi_5 \xi_6 + \xi_5' \xi_6') + (\xi_1 \xi_2' \xi_3 + \xi_1' \xi_4 \xi_4' \xi_5) + (\xi_1 \xi_2' \xi_3' + \xi_1' \xi_4 \xi_4' \xi_6)) \\ &\quad + \xi_1' \xi_4 (\xi_1 \xi_5 (\xi_2 + \xi_2' \xi_3') (\xi_4 \xi_5' + \xi_4' \xi_6) + \xi_1 \xi_2' \xi_3 \xi_5' (\xi_4 \xi_5 + \xi_4' \xi_6') + \xi_1' \xi_4 \xi_4' \xi_5 \xi_5') \\ &\quad + \xi_1' \xi_4' (\xi_1 \xi_6 (\xi_2 + \xi_2' \xi_3) (\xi_4 \xi_5 + \xi_4' \xi_6') + \xi_1 \xi_2' \xi_3 \xi_6' (\xi_4 \xi_5' + \xi_4' \xi_6) + \xi_1' \xi_4 \xi_4' \xi_6 \xi_6'))), \end{aligned} \quad (2)$$

with $\xi_i' = 1 - \xi_i$. Note that the integration domain for three-loop selfenergy diagram, I_{3loop} , can be changed by the similar way.

In order to evaluate I_{2loop} and I_{3loop} , we used DE [3] iteratively for the multi-dimensional integration. Computation time for the numerical integration depends on the total number of evaluation points, N_{total} . In order to take advantage of the hardware characteristics of GRAPE9-MPX, we divide N_{total} into two parts, i -loop (outer loop) part and j -loop (inner loop) part, so as to satisfy $N_{total} = N_i \times N_j$. This is realized by *loop fusion* technique, in which the outer 3-dimensional integration in Eq. (1) can be performed in i -loop and the most inner 3-dimensions in j -loop. In the performance measurements, we set N_i to be 2^{18} for two-loop crossed box and 2^{23} three-loop selfenergy, respectively. On the other hand, we vary $N_j = 2^k$ where $k = 4, 5, 6, \dots, 13$. Hereafter, we call the maximum problem size for $N_j = 2^{13}$. In the maximum problem size, N_{total} is 2^{31} for the two-loop crossed box, and 2^{36} for the three-loop selfenergy, respectively.

The numerical results in quadruple precision obtained by GRAPE9-MPX in the maximum problem size are $I_{2loop} = 0.008536$ and $I_{3loop} = 0.279609$, which are within the relative errors

of 10^{-3} for the two-loop crossed box and 10^{-5} for the three-loop selfenergy compared to the previous works [12, 13]. It seems that the error is slightly larger because of smaller number of N_{total} , but the calculation with this error level is achieved in a very short time, as seen later.

4. Performance results

We measured the performance of GRAPE9-MPX with up to 4 FPGA boards. It is connected a Linux PC (Scientific Linux 6.5) with Intel Xeon 2687W (3.4GHz) CPU on ASRock Xtreme11 motherboard (X79 chipset). The performance results are plotted in Fig.4. From this figure, we can see that for the two-loop crossed box integral, the effective performance in the maximum problem size was 2.4 Gflops with a single board, 4.7 Gflops with 2 boards, and 9.1 Gflops with 4 boards, respectively. On the other hand, the effective performance for the three-loop selfenergy integral in the maximum problem size was 8.7 Gflops with 4 boards.

This figure also shows that the speed up by multiple boards is well scalable in the maximum problem size. It becomes 3.8 times faster using 4 boards, and 1.9 times faster using 2 boards than using a single board. On the other hand, the scalability is not good in smaller N_j . This is because the overhead by communication between the GRAPE9-MPX boards and the host computer via PCI express is not negligible for smaller N_j .

We compared the efficiency of the effective performance for the two-loop crossed box integral and for the three-loop selfenergy integral. We show that it is about 34% of the theoretical peak performance for the two-loop crossed box, whereas about 33% for the three-loop selfenergy. The reason why the latter is slightly lower than the former comes from the difference of the number of instructions. From LSUMP, the operation count of the instruction is 91 for two-loop crossed box, whereas it is 89 for three-loop selfenergy.

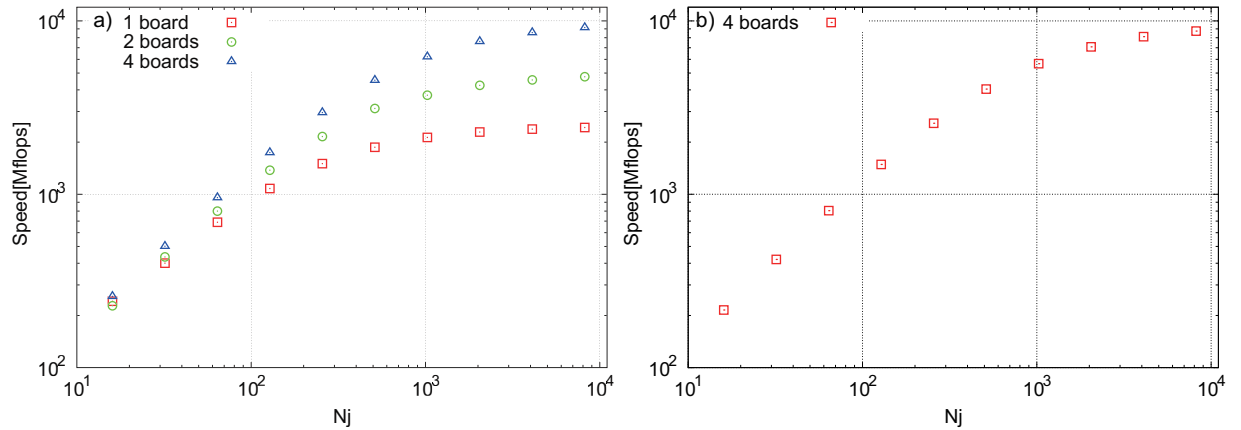


Figure 4. Performance results for two-loop crossed box diagram (left) and three-loop selfenergy diagram (right).

We also compared the computation time for the two-loop integral in the maximum problem size measured in GRAPE9-MPX of MP4 with 4boards, and the time measured in software implementations with the same numerical format (15-bit exponent and 112-bit mantissa). The measured calculation time is 21.3 sec for GRAPE9-MPX, 1074.7 sec for `quadmath` on gcc4.6.3, and 6086.7 sec for `GMP-MPFR`. Although we used `OpenMP` for the multi-threaded computing and the number of threads used was 16, GRAPE9-MPX system was 52.9 times faster than `quadmath` and, 319.5 times faster than `GMP-MPFR`. Thus, we could show the advantage of our system in the performance.

5. Summary

We have developed GRAPE9-MPX system, an accelerator system for multi-precision arithmetic operations which can accelerate the calculation of Feynman loop integrals. Our system consists of multiple FPGA boards in which a lot of PEs are implemented and forms SIMD way. Each PE has a dedicated logic for multi-precision operations. We measured the performance of the system in quadruple precision (15-bit exponent and 112-bit mantissa) for the two-loop crossed box and the three-loop selfenergy integrals, and got the effective performance of 9.1 Gflops and of 8.7 Gflops with 4 boards in the maximum program size, respectively.

One of the advantages of our system is that we have developed not only the hardware with high performance in multi-precision calculation, but also the programming interface which enables us to port applications easily for our system. The combination of Goose and LSUMP provides a simple way to use GRAPE9-MPX. All we have to do is to insert the directive in the original code. This simple programming model plays an essential role in treating various kinds of loop integrals in the automatic computation of Feynman diagrams.

Although we mainly presented the results of MP4 in this paper, we have measured the performance of MP6 and MP8 for the same Feynman loop integral. For the case of the two-loop crossed box, we achieved 4.4 Gflops for MP6 and 2.3 Gflops for MP8 with 4 boards, respectively. The detail of the results will appear in near future.

Acknowledgment

We acknowledge the support from Grants-in-Aid for Scientific Research 23540328 and 24540292.

References

- [1] de Doncker E, Shimizu Y, Fujimoto J and Yuasa F 2004 *Computer Physics Communications* **159** 145–156
- [2] Yuasa F, de Doncker E, Fujimoto J, Hamaguchi N, Ishikawa T and Shimizu Y 2007 *PoS(ACAT)* vol 087 URL <http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=50>
- [3] Mori M 2005 *Publications of the Research Institute for Mathematical Sciences* **41**(4) 897–935
- [4] GMP URL <https://gmplib.org/>
- [5] MPFR URL <http://www.mpfr.org/>
- [6] High-Precision Software Directory URL <http://crd-legacy.lbl.gov/~dhbailey/mpdist/>
- [7] Koike K 2009 *Development of Control Processor for Multipurpose Computer GRAPE-DR and its Performance Evaluation* Thesis Graduate University for Advanced Studies URL <http://id.nii.ac.jp/0201/00001514>
- [8] Daisaka H, Nakasato N, Makino J, Yuasa F and Ishikawa T 2011 *Procedia Computer Science* **4** 878–887
- [9] Nakasato N, Daisaka H, Fukushige T, Kawai A, Makino J, Ishikawa T and Yuasa F 2012 *Embedded Multicore Socs (MCSoc), 2012 IEEE 6th International Symposium on* 75–83
- [10] Altera Co *Arria V Device Handbook* URL www.altera.co.jp/literature/hb/arria-v/arriav_handbook.pdf
- [11] Nakasato N and Makino J 2009 *IEEE International Conference on Cluster Computing and Workshops* pp 1–9
- [12] Yuasa F, de Doncker E, Hamaguchi N, Ishikawa T, Kato K, Kurihara Y and Fujimoto J 2012 *Computer Physics Communications* **183** 2136–2144
- [13] Laporta S 2000 *International Journal of Modern Physics A* **15** 5087